MAC 21.02.2024

(curs 1 - S 1)

1. Fundamente

Operatiile aritmetice fundamentale : adunarea si inmultirea.
Sunt necesare pentru a evalua un polinom P(x) intr-un punct x.

1.1. Evaluarea unui polinom:

$P(x) = 2x^4 + 3x^3 - 3x^2 + 5x - 1, \quad x = \frac{1}{2}$

I. Metoda directă

$$P\left(\frac{1}{2}\right) = 2*\frac{1}{2}*\frac{1}{2}*\frac{1}{2}*\frac{1}{2}+3*\frac{1}{2}*\frac{1}{2}*\frac{1}{2}-3*\frac{1}{2}*\frac{1}{2}+5*\frac{1}{2}-1 = \frac{5}{4}$$

Numarul de înmultiri necesare este 10, la care se adauga 4 adunari

II. Găsim puterile numărului $\frac{1}{2}$

$$\begin{aligned}\frac{1}{2}*\frac{1}{2} &= \left(\frac{1}{2}\right)^2\\ \left(\frac{1}{2}\right)^2*\frac{1}{2} &= \left(\frac{1}{2}\right)^3\\ \left(\frac{1}{2}\right)^3*\frac{1}{2} &= \left(\frac{1}{2}\right)^4.\end{aligned}$$

- acum, putem face o simplă însumare a termenilor:

$$P\left(\frac{1}{2}\right) = 2*\left(\frac{1}{2}\right)^4+3*\left(\frac{1}{2}\right)^3-3*\left(\frac{1}{2}\right)^2+5*\frac{1}{2}-1=\frac{5}{4}.$$

- avem 3 înmulţiri ale lui $1/2$, plus alte 4 înmulţiri
- am redus numărul total de înmulţiri la 7, cu aceleaşi 4 adunări
- este reducerea de la 14 la 11 operaţii o îmbunătăţire semnificativă?
- dacă polinomul trebuie evaluat pentru diferite valori ale lui x, de mai multe ori pe secundă, atunci diferenţa poate fi esenţială

III. Înmulțirea imbricată (metoda lui Horner)

$$\begin{aligned}P(x) &= -1 + x(5 - 3x + 3x^2 + 2x^3) =\\ &= -1 + x(5 + x(-3 + 3x + 2x^2)) =\\ &= -1 + x(5 + x(-3 + x(3 + 2x)))\end{aligned}$$

Evaluarea se face dinspre interior spre exterior.

$$\begin{aligned}&\text{înmulţim } \frac{1}{2}*2, && \text{adunăm } +3 \to 4\\ &\text{înmulţim } \frac{1}{2}*4, && \text{adunăm } -3 \to -1\\ &\text{înmulţim } \frac{1}{2}*-1, && \text{adunăm } +5 \to \frac{9}{2}\\ &\text{înmulţim } \frac{1}{2}*\frac{9}{2}, && \text{adunăm } -1 \to \frac{5}{4}.\end{aligned}$$

In general, un polinom de gradul d poate fi evaluat utilizând aceasta metoda folosind d înmultiri și d adunari.

Forma generală a unui polinom

$$c_1 + c_2 x + c_3 x^2 + c_4 x^3 + c_5 x^4 = c_1 + x(c_2 + x(c_3 + x(c_4 + x(c_5))))$$

Pentru interpolare, vom avea nevoie de forma:

$$c_1 + (x - r_1)(c_2 + (x - r_2)(c_3 + (x - r_3)(c_4 + (x - r_4)(c_5))))$$

unde r_1, r_2, r_3, r_4 - puncte de bază

Exemplul 1

- găsiţi o metodă eficientă pentru evaluarea polinomului $P(x) = 4x^5 + 7x^8 - 3x^{11} + 2x^{14}$
- ideea este de a factoriza x^5 din fiecare termen şi de a scrie polinomul în x^3:

$$\begin{aligned} P(x) &= x^5(4 + 7x^3 - 3x^6 + 2x^9) \\ &= x^5 * (4 + x^3 * (7 + x^3 * (-3 + x^3 * (2)))). \end{aligned}$$

- pentru fiecare punct x, trebuie să calculăm mai întâi $x * x = x^2$, $x * x^2 = x^3$, şi $x^2 * x^3 = x^5$
- aceste trei înmulţiri, împreună cu înmulţirea lui x^5, şi cele trei înmulţiri şi trei adunări necesare evaluării polinomului de gradul 3 în x^3 dau numărul total de operaţii necesar evaluării acestui polinom, şi anume 7 înmulţiri şi 3 adunări

1.2. Numere binare

Un nr în baza 2: $\ldots b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0 + b_{-1} \cdot 2^{-1} + b_{-2} \cdot 2^{-2} \ldots$

$(4)_{10} = (100)_2$

$\left(\frac{3}{4}\right)_{10} = (0,75)_{10} = (0.11)_2$

1.2.1. Conversia din zecimal în binar

$(53.7)_{10} = (53)_{10} + (0.7)_{10}$

Partea întreagă: $(53)_{10} = (110101)_2$

Partea fracţionară:

0.7×2	$0.4 + 1$
0.4×2	$0.8 + 0$
0.8×2	$0.6 + 1$
0.6×2	$0.2 + 1$
0.2×2	$0.4 + 0$
0.4×2	$0.8 + 0$

Observăm că acest proces se repetă la infinit după 4 pași.

$$\Rightarrow (0.7)_{10} = (0.1\underline{0110}0110\,0110\ldots)_2 = (0.1\overline{0110})_2$$

$$\Rightarrow (53.7)_{10} = (110101.1\overline{0110})_2$$

1.2.2. Conversia din binar în zecimal

$$(10101)_2 = 1\cdot 2^4 + 0\cdot 2^3 + 1\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0 = (21)_{10}$$

$$(0.1011)_2 = 1\cdot \frac{1}{2} + 0\cdot \frac{1}{2^2} + 1\cdot \frac{1}{2^3} + 1\cdot \frac{1}{2^4} = \left(\frac{11}{16}\right)_{10}$$

% $(0.\overline{1011})_2$

Înmulțim nr. cu 2^4 => deplasare la stânga cu 4 poziții

$$2^4 \cdot x = 1011.\overline{1011}$$

$$x = 0000.\overline{1011}$$

$$\Rightarrow x(2^4-1) = (1011)_2 = (11)_{10}$$

$$\Rightarrow x = (0.\overline{1011})_2 = \left(\frac{11}{15}\right)_{10}$$

% $x = (0.10\overline{101})_2$

$$y = 2^2 \cdot x = (10.\overline{101})_2$$

$$\{y\} = (0.\overline{101})_2 = z$$

$$\Rightarrow 2^3 \cdot z = (101.\overline{101})_2$$

$$z = (000.\overline{101})_2$$

$$\Rightarrow z(2^3-1) = (101)_2 = 5 \Rightarrow z = \frac{5}{7}$$

$$\Rightarrow y = (10)_2 + \frac{5}{7} = 2 + \frac{5}{7} = \frac{19}{7}$$

$$\Rightarrow x = 2^{-2} \cdot y = \left(\frac{19}{28}\right)_{10}$$

1.3. Reprezentarea in virgula flotanta a numerelor reale

1.3.1. Formate de virgulă flotantă

Un număr în virgulă flotantă este format din trei părți:

a) semnul (+ sau -)

b) mantisa (care conține șirul de biți semnificativi)

c) exponentul

Există 3 nivele de precizie:

		BIȚI ALOCAȚI
I.	simplă	→ 32
II.	dublă	→ 64
III.	extinsă (lungă)	→ 80

precizia	semnul	exponentul	mantisa
simplă	1	8	23
dublă	1	11	52
dublă lungă	1	15	64

Definiţia 1

Numărul **epsilon maşină**, notat cu ϵ_{mach}, este distanţa dintre 1 şi cel mai mic număr în virgulă flotantă mai mare decât 1. Pentru standardul IEEE de virgulă flotantă în dublă precizie, avem că

$$\epsilon_{\text{mach}} = 2^{-52}.$$

- numărul zecimal $9.4 = (1001.\overline{0110})_2$ este aliniat la stânga astfel:

$$+1.\boxed{0010110011001100110011001100110011001100110011001100}110\ldots\times 2^3,$$

unde am pus într-un chenar primii 52 de biţi ai mantisei

1.3.1 Formate de virgulă flotantă

Algoritmul 1 (Regula IEEE a rotunjirii la cea mai apropiată valoare)

Pentru dubla precizie, dacă al 53-lea bit de la dreapta virgulei binare este 0, atunci rotunjim în jos (trunchiem după bitul 52). Dacă bitul al 53-lea este 1, atunci rotunjim în sus (adunăm 1 la bitul 52), cu excepţia cazului în care biţii de după 1 sunt 0, caz în care 1 este adunat la bitul 52 dacă şi numai dacă bitul 52 este 1.

- pentru numărul 9.4 discutat anterior, cel de-al 53-lea bit din dreapta virgulei binare este 1 şi este urmat de alţi biţi care nu sunt toţi zero
- conform regulii rotunjirii la cea mai apropiată valoare, trebuie să facem o rotunjire în sus, adică să adunăm un 1 la bitul 52
- prin urmare, numărul în virgulă flotantă care îl reprezintă pe 9.4 este

$$+1.\boxed{0010110011001100110011001100110011001100110011001101}\times 2^3. \quad (7)$$

Definiţia 2

Notăm numărul în virgulă flotantă în dublă precizie IEEE asociat lui x, folosind regula rotunjirii la cea mai apropiată valoare, cu **fl(x)**.

- în operaţiile aritmetice realizate în calculator, numărul real x este înlocuit cu şirul de biţi $\mathrm{fl}(x)$
- conform definiţiei de mai sus, fl(9.4) este numărul în reprezentare binară dat de (7)
- am ajuns la această reprezentare în virgulă flotantă înlăturând partea infinită dată de $0.\overline{1100} \times 2^{-52} \times 2^{3} = 0.\overline{0110} \times 2^{-51} \times 2^{3} = 0.4 \times 2^{-48}$ din capătul din dreapta al numărului, şi apoi adunând $2^{-52} \times 2^{3} = 2^{-49}$ în pasul de rotunjire
- prin urmare,

$$\begin{aligned} \mathrm{fl}(9.4) &= 9.4 + 2^{-49} - 0.4 \times 2^{-48} \\ &= 9.4 + (1-0.8)2^{-49} \\ &= 9.4 + 0.2 \times 2^{-49}. \end{aligned} \tag{8}$$

- cu alte cuvinte, un calculator care foloseşte reprezentarea în dublă precizie şi regula rotunjirii la cea mai apropiată valoare face o eroare de 0.2×2^{-49} atunci când stochează numărul 9.4
- vom spune că 0.2×2^{-49} este **eroarea de rotunjire**
- ceea ce trebuie reţinut este faptul că numărul în virgulă flotantă care îl reprezintă pe 9.4 nu este egal cu 9.4, deşi este foarte aproape de această valoare
- pentru a cuantifica această apropiere, vom folosi definiţia standard a erorii

Definiţia 3

Fie x_c o versiune calculată a valorii exacte x. Atunci

$$\textbf{eroarea absolută} = |x_c - x|,$$

şi

$$\textbf{eroarea relativă} = \frac{|x_c - x|}{|x|},$$

dacă această din urmă cantitate există ($x \neq 0$).

Algoritmul 2 (Eroarea relativă de rotunjire)

În cadrul standardului IEEE, eroarea relativă de rotunjire $\mathrm{fl}(x)$ este mai mică decât jumătate din numărul epsilon maşină:

$$\frac{|\mathrm{fl}(x) - x|}{|x|} \leq \frac{1}{2}\epsilon_{\mathrm{mach}}. \tag{9}$$

- în cazul numărului $x = 9.4$, am găsit eroarea de rotunjire în (8), care trebuie să satisfacă (9):

$$\frac{|\mathrm{fl}(9.4) - 9.4|}{9.4} = \frac{0.2 \times 2^{-49}}{9.4} = \frac{8}{47} \times 2^{-52} < \frac{1}{2}\epsilon_{\mathrm{mach}}.$$

Exemplul 2

- găsiţi reprezentarea în dublă precizie $fl(x)$ şi eroarea de rotunjire pentru $x = 0.4$
- deoarece $(0.4)_{10} = (0.\overline{0110})_2$, alinierea la stânga a acestui număr binar ne dă:

$$\begin{aligned} 0.4 &= 1.10\overline{0110} \times 2^{-2} \\ &= +1.\boxed{1001100110011001100110011001100110011001100110011001} \\ &\quad 100110\ldots \times 2^{-2}. \end{aligned}$$

- prin urmare, în conformitate cu regula de rotunjire, fl(0.4) este

$$+1.\boxed{1001100110011001100110011001100110011001100110011010} \times 2^{-2}.$$

- aici, 1 a fost adunat la bitul 52, ceea ce a determinat o schimbare şi a bitului 51, ca urmare a transportului din adunarea binară
- analizând cu atenţie, am înlăturat $2^{-53} \times 2^{-2} + 0.\overline{0110} \times 2^{-54} \times 2^{-2}$ în cadrul trunchierii şi am adunat $2^{-52} \times 2^{-2}$ prin rotunjirea în sus

1.3.1 Formate de virgulă flotantă

- prin urmare,

$$\begin{aligned} fl(0.4) &= 0.4 - 2^{-55} - 0.4 \times 2^{-56} + 2^{-54} \\ &= 0.4 + 2^{-54}(-1/2 - 0.1 + 1) \\ &= 0.4 + 2^{-54}(0.4) \\ &= 0.4 + 0.1 \times 2^{-52}. \end{aligned}$$

- observăm că eroarea relativă de rotunjire pentru 0.4 este $0.1/0.4 \times \epsilon_{\text{mach}} = 1/4 \times \epsilon_{\text{mach}}$, conform cu (9)

1.3.2. Reprezentarea numerelor în calculator

fiecărui număr în virgulă flotantă în dublă precizie îi este asignat un cuvânt de 8 octeţi (1 octet = 8 biţi), sau 64 de biţi, pentru a stoca cele trei părţi ale sale

fiecare astfel de cuvânt are forma

$$\boxed{se_1e_2\ldots e_{11}b_1b_2\ldots b_{52}}, \tag{10}$$

unde mai întâi este stocat semnul, urmat de 11 biţi reprezentând exponentul şi de 52 de biţi care urmează virgulei zecimale, care reprezintă mantisa

bitul de semn s este 0 pentru un număr pozitiv şi 1 pentru un număr negativ

cei 11 biţi reprezentând exponentul vin din numărul binar pozitiv care rezultă din adunarea lui $2^{10} - 1 = 1023$ la exponent, cel puţin pentru exponenţi între -1022 şi 1023

- aceasta acoperă valorile lui $e_1 \dots e_{11}$ de la 1 la 2046, lăsând valorile 0 şi 2047 pentru scopuri speciale, care vor fi detaliate ulterior
- numărul 1023 se numeşte **biasul exponent** al formatului în dublă precizie
- este folosit pentru a converti atât exponenţii pozitivi cât şi pe cei negativi în numere binare pozitive pentru a fi stocate în biţii de exponent
- pentru precizia simplă şi dublă lungă, valorile biasului exponent sunt 127 şi respectiv 16383
- formatul hexazecimal constă din exprimarea celor 64 de biţi ai reprezentării în virgulă flotantă (10) ca 16 numere hexazecimale
- prin urmare, primele 3 numerale hexazecimale reprezintă semnul şi exponentul combinate, în timp ce ultimele 13 conţin mantisa

Exemplul 3

- găsiţi reprezentarea în virgulă flotantă în dublă precizie a numărului real 9.4
- din (7), avem că semnul este $s = 0$, exponentul este 3, şi cei 52 de biţi ai mantisei de după virgula zecimală sunt

$$\boxed{0010}\boxed{1100}\boxed{1100}\boxed{1100}\boxed{1100}\boxed{1100}\boxed{1100}\boxed{1100}\boxed{1100}\boxed{1100}\boxed{1100}\boxed{1100}\boxed{1101} \to (2CCCCCCCCCCCD)_{16}.$$

- adunând 1023 la exponent obţinem $1026 = 2^{10} + 2$, sau $(10000000010)_2$
- combinaţia semn – exponent este $(010000000010)_2 = (402)_{16}$, dând formatul hexazecimal $4022CCCCCCCCCCCD$

- ne întoarcem acum la valorile speciale ale exponentului 0 şi 2047
- ultima, 2047, este folosită pentru a reprezenta ∞ dacă şirul de biţi ai mantisei este format doar din zerouri şi `NaN`, adică Not a Number, altfel
- deoarece 2047 este reprezentat prin unsprezece biţi de 1, sau $e_1 e_2 \dots e_{11} = (111\,1111\,1111)_2$, primii doisprezece biţi ai lui `Inf` şi `-Inf` sunt $\boxed{0111}\boxed{1111}\boxed{1111}$ şi, respectiv, $\boxed{1111}\boxed{1111}\boxed{1111}$, şi cei 52 de biţi rămaşi (mantisa) sunt zero
- numărul `NaN` începe de asemenea cu $\boxed{1111}\boxed{1111}\boxed{1111}$ dar are o mantisă nenulă
- putem rezuma cele de mai sus în tabelul următor:

numărul	exemplu	format hexazecimal
`+Inf`	$1/0$	$7FF0000000000000$
`-Inf`	$-1/0$	$FFF0000000000000$
`NaN`	$0/0$	$FFFxxxxxxxxxxxxx$

unde x-urile denotă biţi care nu sunt toţi egali cu zero

- exponentul special 0, adică $e_1 e_2 \ldots e_{11} = (000\,0000\,0000)_2$, denotă de asemenea o abatere de la forma standard a unui număr în virgulă flotantă
- în acest caz, numărul este interpretat ca numărul în virgulă flotantă nenormalizat

$$\pm 0.\boxed{b_1 b_2 \ldots b_{52}} \times 2^{-1022}. \tag{11}$$

- adică, *doar în acest caz, bitul cel mai din stânga nu este presupus a fi egal cu* 1
- aceste numere nenormalizate se numesc numere în virgulă flotantă **subnormale**
- ele extind gama numerelor foarte mici cu încă câteva ordine de mărime
- prin urmare, $2^{-52} \times 2^{-1022} = 2^{-1074}$ este cel mai mic număr diferit de zero care este reprezentabil în dublă precizie
- cuvântul corespunzător care va fi stocat în calculator este

$$\boxed{0\,|\,00000000000\,|\,0000000000000000000000000000000000000000000000000001}.$$

- trebuie făcută distincţia între cel mai mic număr reprezentabil 2^{-1074} şi $\epsilon_{\text{mach}} = 2^{-52}$
- sunt multe numere mai mici decât ϵ_{mach} care sunt reprezentabile, chiar dacă adunându-le la 1 nu vom obţine niciun efect
- pe de altă parte, numerele în dublă precizie mai mici decât 2^{-1074} nu pot fi reprezentate deloc
- numerele subnormale includ şi cel mai important număr, 0
- de fapt, reprezentarea subnormală include două numere în virgulă flotantă diferite, $+0$ şi -0, care sunt tratate în calcule ca fiind acelaşi număr real
- reprezentarea în virgulă flotantă a lui $+0$ are bitul de semn $s = 0$, biţii de exponent $e_1 \ldots e_{11} = 00000000000$, şi mantisa 52 de zerouri; pe scurt, toţi cei 64 de biţi sunt zero
- formatul hexazecimal pentru $+0$ este 0000000000000000
- pentru numărul -0, totul este exact la fel, cu excepţia bitului de semn $s = 1$
- formatul hexazecimal pentru -0 este 8000000000000000

1.3.3 Adunarea numerelor în virgulă flotantă

- adunarea în calculator constă din alinierea virgulelor zecimale ale celor două numere de adunat, adunarea lor, şi stocarea rezultatului tot ca un număr în virgulă flotantă
- adunarea propriu-zisă poate fi realizată cu precizie mai mare (cu mai mult de 52 de biţi) deoarece are loc într-un registru dedicat special pentru acest scop
- după adunare, rezultatul trebuie rotunjit înapoi la 52 de biţi după virgula binară pentru a fi stocat ca un număr în virgulă flotantă
- de exemplu, adunarea lui 1 la 2^{-53} va apărea după cum urmează:

$$\begin{aligned}
& 1.\boxed{00\ldots0} \times 2^0 + 1.\boxed{00\ldots0} \times 2^{-53} \\
= \quad & 1.\boxed{0000000000000000000000000000000000000000000000000000} \times 2^0 \\
+ \quad & 0.\boxed{0000000000000000000000000000000000000000000000000000}1 \times 2^0 \\
= \quad & 1.\boxed{0000000000000000000000000000000000000000000000000000}1 \times 2^0.
\end{aligned}$$

- acest număr este stocat ca $1.0 \times 2^0 = 1$, conform regulii de rotunjire
- prin urmare, $1 + 2^{-53}$ este egal cu 1 în aritmetica în dublă precizie IEEE

Exemplul 4

- găsiţi suma în virgulă flotantă în dublă precizie $(1 + 3 \times 2^{-53}) - 1$
- bineînţeles, în aritmetica reală, răspunsul este 3×2^{-53}
- însă calculele în aritmetica în virgulă flotantă pot da un rezultat diferit
- observăm că $3 \times 2^{-53} = 2^{-52} + 2^{-53}$
- prima adunare este

$$
\begin{aligned}
& 1.\boxed{00\ldots0} \times 2^0 + 1.\boxed{10\ldots0} \times 2^{-52} \\
= \quad & 1.\boxed{0000000000000000000000000000000000000000000000000000} \times 2^0 \\
+ \quad & 0.\boxed{0000000000000000000000000000000000000000000000000001}1 \times 2^0 \\
= \quad & 1.\boxed{0000000000000000000000000000000000000000000000000001}1 \times 2^0.
\end{aligned}
$$

- acesta este cazul de excepţie pentru regula rotunjirii
- deoarece bitul 52 din sumă este 1, trebuie să facem o rotunjire în sus, ceea ce înseamnă să adunăm 1 la bitul 52
- după transport, obţinem

$$1.\boxed{0000000000000000000000000000000000000000000000000010} \times 2^0,$$

  care este reprezentarea lui $1 + 2^{-51}$
- prin urmare, după ce scădem 1, rezultatul va fi 2^{-51}, care este egal cu $2\epsilon_{\text{mach}} = 4 \times 2^{-53}$
- din nou, se poate observa diferenţa dintre aritmetica în calculator şi aritmetica exactă

2. Rezolvarea ecuatiilor

Def 1: Funcția $f(x)$ are o rădăcină în $x = r$ dacă $f(r) = 0$

Def 2: r este un punct fix al funcției g dacă $g(r) = r$

Proprietate: Orice ecuație $f(x) = 0$ poate fi transformată într-o problemă de punct fix.

ex: $x^3 + x - 1 = 0$

1) $x^3 + x - 1 = 0 \;\big|\; +1 - x \Rightarrow x = 1 - x^3 = g(x)$

2) $x^3 + x - 1 = 0 \;\big|\; +1 - x \Rightarrow x^3 = 1 - x \Rightarrow x = \sqrt[3]{1-x} = g(x)$

3) $x^3 + x - 1 = 0 \;\big|\; +2x^3 + 1 \Rightarrow 3x^3 + x = 2x^3 + 1$

$\Rightarrow x(3x^2 + 1) = 2x^3 + 1$

$\Rightarrow x = \dfrac{2x^3 + 1}{3x^2 + 1} = g(x)$

2.1. Metoda bisecției $\to f(x)=0$

2.1.1. Găsirea limitelor între care se află o rădăcină

$\to$ verificăm dacă $\exists$ o rădăcină

$[a, b]$

$$\left.\begin{array}{l} f(a)\cdot f(b) < 0 \\ f - \text{continuă și } r \in [a, b] \end{array}\right\} \Rightarrow f(r)=0$$

Teorema 1 (Teorema valorii intermediare)

Fie f o funcţie continuă pe intervalul $[a, b]$. Atunci f ia orice valoare între $f(a)$ şi $f(b)$. Mai precis, dacă y este un număr între $f(a)$ şi $f(b)$, atunci există un număr c care satisface $a \leq c \leq b$ astfel încât $f(c) = y$.

Teorema 2

Fie f o funcţie continuă pe $[a, b]$, care satisface $f(a)f(b) < 0$. Atunci f are o rădăcină între a şi b, adică există un număr r care satisface $a < r < b$ şi $f(r) = 0$.

- în Figura 1, $f(0)f(1) = (-1)(1) < 0$
- există o rădăcină imediat la stânga lui 0.7
- cum putem rafina presupunerea iniţială despre locaţia rădăcinii cu mai multe zecimale exacte?

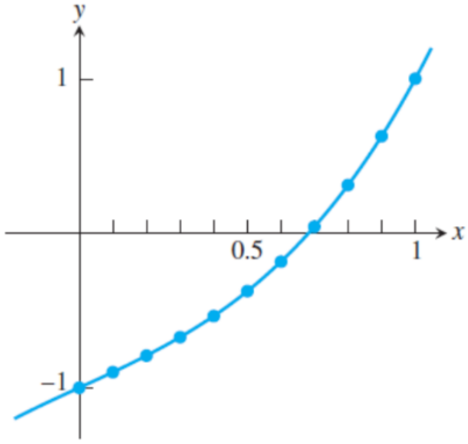

Figura 1: **Graficul funcţiei $f(x) = x^3 + x - 1$.** Funcţia are o rădăcină între 0.6 şi 0.7.

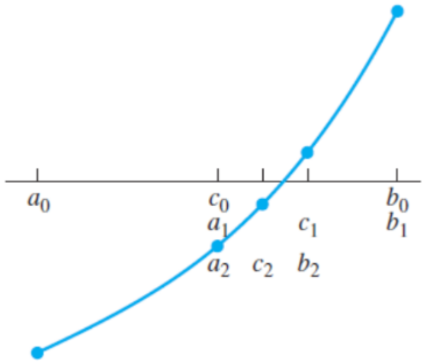

Figura 2: **Metoda bisecţiei.** În primul pas, semnul lui $f(c_0)$ este verificat. Deoarece $f(c_0)f(b_0) < 0$, asignăm $a_1 = c_0$, $b_1 = b_0$, şi intervalul este înlocuit cu jumătatea lui dreaptă $[a_1, b_1]$. În al doilea pas, subintervalul este înlocuit cu jumătatea lui stângă $[a_2, b_2]$.

Algoritmul 3 (**Metoda bisecţiei**)

Dându-se un interval iniţial $[a, b]$ astfel încât $f(a)f(b) < 0$
while $(b - a)/2 >$ TOL
  $c = (a + b)/2$
  if $f(c) = 0$, **stop**, **end**
  if $f(a)f(c) < 0$
    $b = c$
  else
    $a = c$
  end
end
Intervalul final $[a, b]$ conţine o rădăcină.
Aproximarea rădăcinii este $(a + b)/2$.

```
function x = metoda_bisectiei(f,a, b, tol)

if f(a) * f(b) >= 0
    error('Conditia f(a) * f(b) < 0 nu este satisfacuta!')
end

while (b - a) / 2 > tol
    c = (a + b) / 2;
    if f(c) == 0
        break;
    end
    if f(a) * f(c) < 0
        b = c;
    else
        a = c;
    end
end

x = (a + b) /2;
```